TECH-X CORPORATION

GPULib

Version 1.6.2

This QuickStartGuide describes the installation and use of GPULib on Windows systems. For Mac and Linux systems, please refer to the README.txt file.

Overview

GPULib is a library of vector operations to take advantage of the processing power available in present and future graphics cards.

Due to the fact that GPU technology is rapidly advancing and there are myriad variations in hardware, it is possible that you will encounter issues with this software, but if so, please do not hesitate to contact our support team (support@txcorp.com). Our staff is working diligently with customers to alleviate all issues as they arise. We highly value your feedback and would like to hear suggestions for additional kernels, high-level routines or comments in general. Unfortunately, not all language bindings offer complete support for all features of GPULib. The IDL bindings are the most complete.

Step 1: Check your graphics card

GPULib is built on top of NVIDIA CUDA (see "What is CUDA?" at http://www.nvidia.com/object/what_is_cuda_new.html), so you will need to have a CUDA-enabled graphics card before proceeding. Make sure your card is supported by checking the model against the list at

http://www.nvidia.com/object/cuda_gpus.html

Keep in mind that some cards are supported only with an appropriate amount of memory. If your card is on the list, but does not have enough memory, do not expect it to work with GPULib.

Step 2: Check your graphics drivers

Make sure you have the latest drivers for your graphics card by reviewing the following website:

http://www.nvidia.com/drivers

Step 3: Install CUDA

Visit the CUDA Downloads page at:

http://developer.nvidia.com/object/cuda_downloads.html

(At the time of this writing, the above link points to the 5.5 version of the CUDA Toolkit.)

Be sure to download the appropriate files for your operating system (Windows XP, Vista, or Windows 7), computer type (desktop or notebook), and CPU (32-bit or 64-bit). Detailed instructions on installing the CUDA components are provided at the NVIDIA Web site.

Step 4: Install GPULib

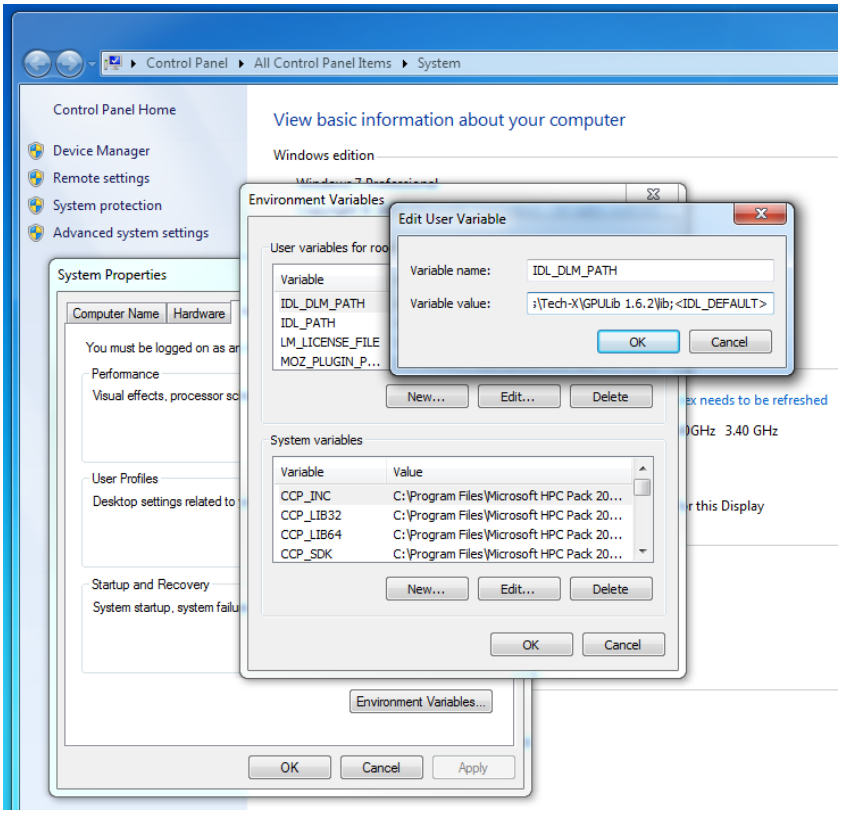
Now you are ready to install GPULib. Double-click the installer executable to run it:

`GPULib-1.6.2-Win64.exe`

Select the option to add GPULib to the path for either the current user or all users.
After running the installer, the pre-built IDL Dynamic Loadable Module (DLM) is located in the lib directory under the installation directory selected by the users (default is C:\Program Files\Tech-X\GPULib 1.6.2\lib). To use GPULib in IDL, you will also need to modify your IDL environment. For example, we recommend defining the following as user environment variables:

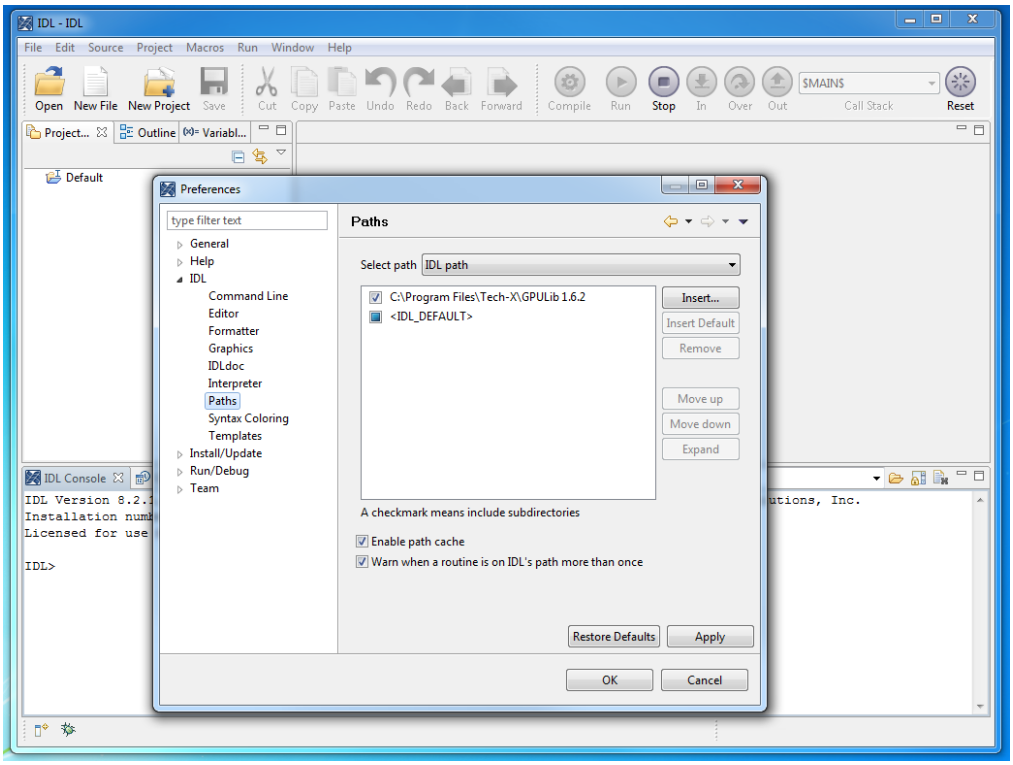
```
Variable name: IDL_PATH
Variable value: +C:\Program Files\Tech-X\GPULib 1.6.2;<IDL_DEFAULT>

Variable name: IDL_DLM_PATH
Variable value: +C:\Program Files\Tech-X\GPULib 1.6.2\lib;<IDL_DEFAULT>
```

These are set by bringing up the Window System Properties dialog, clicking the "Environment Variables" button, and clicking the "New" button under "User variables". The associated dialogs should look similar to the screen capture below:



The "+" signs indicate a recursive search and are necessary to find all of the PRO files distributed with GPULib. A fresh command prompt window will be necessary after setting environment variables.

Once these variables are set, they can be verified in the IDL Workbench. From the "Window" menu in the Workbench, choose the "Preferences" item. Then from the Preference dialog window, choose "IDL -> Paths". There will be a pull-down menu in the upper right that allows you to select "IDL path" or "DLM path" to check that the system variables are picked up by IDL. Make sure that the checkbox for each path is ticked, which indicates subdirectories will be included.



As a test of the installation, from inside the IDL Workbench, enter the command

```
IDL> @gpu_test
```

You can also enter the following command in a Command Prompt window:

```
"C:\Program Files\Exelis\IDL82\bin\bin.x86_64\idl.exe" gpu_run_test
```

(if your IDL is located in a non-default location, then this command will need to be altered accordingly). Your output should look something like...

```
   IDL Version 8.2.1, Microsoft Windows (Win32 x86 64 m64). (c) 2012, Exelis Visual
Information Solutions, Inc.
% Loaded DLM: GPULIB.
GPULib 1.6.2 (Revision: 2541)
Graphics card: GeForce GT 620, compute capability: 2.1, memory: 1814 MB available, 2048
MB total
CUDA version: 5.5
MAGMA version: 1.4.0
Checking GPU memory allocation...cudaSuccess

% Compiled module: GPUVALID.
CPU results:
     0.756607      2.33993     0.196372     0.516154    0.0442747     0.839950
GPU results:
     0.756607      2.33993     0.196372     0.516154    0.0442747     0.839950

CPU Time = 0.873000
GPU Time = 0.078000
Speedup  = 11.2x
```

(Note: Speedup depends on the ratio of GPU to CPU performance. For actual speedup data from sample systems, see https://ice.txcorp.com/trac/GPULib/wiki/SpeedupTable.) If the **`gpu_run_test`** command indicates a negligible speedup (on the order of 1), and/or does not display any information about your graphics card, this may indicate that your IDL paths are not set correctly, or that GPULib is in emulation mode because there is no Cuda-enabled graphics card available. Check your paths, restart your command prompt window, and retry.

Running the Demos

GPULib comes with 9 demos. They can be run from the IDL command prompt. For example:

```
IDL> gpu_swirl_demo
```

A display window with a swirled image should appear (see image below). A 2D interpolation is then done first using the CPU and second using the GPU. You should see the swirl pattern unfold to vertical lines and then back to a swirl for each processor type. The second time through (for the GPU) should be faster.

The full list of demo routine names is:

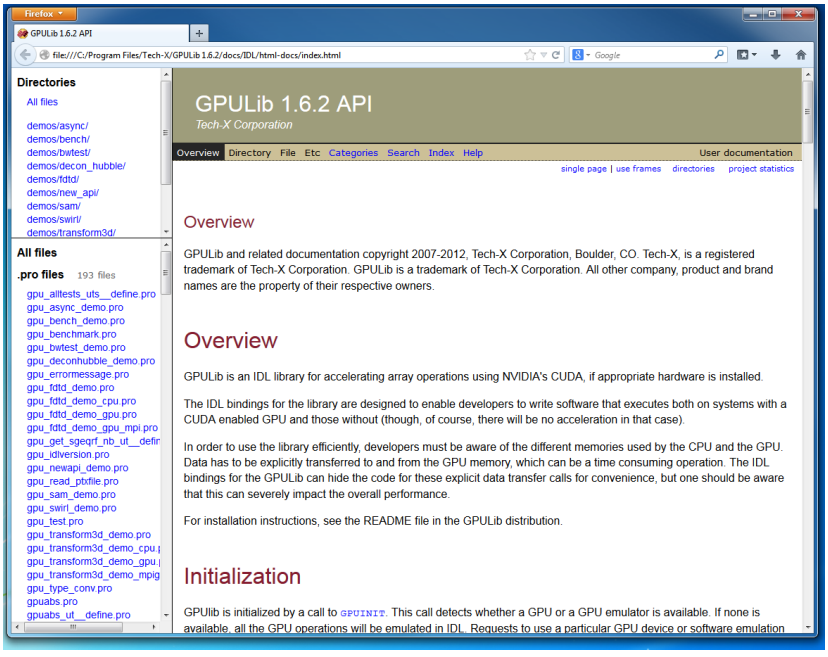
```
gpu_async_demo
gpu_bench_demo
gpu_bwtest_demo
gpu_deconhubble_demo
gpu_fdtd_demo
gpu_newapi_demo
gpu_sam_demo
gpu_swirl_demo
gpu_transform3d_demo
```

The documentation for each demo is contained in the GPULib documentation (see next section).

Documentation of IDL Functions and Demos

The API documentation of the IDL routines included with GPULib can be found in the docs directory in the installation directory (default is C:\Program Files\Tech-X\GPULib 1.6.2\docs). The documentation can be viewed with a web browser starting with the file:

```
C:\Program Files\Tech-X\GPULib 1.6.2\docs\IDL\html-docs\index.html
```



Troubleshooting

Why doesn't one of the demos work?
If you have difficulties running the demos, then you might want to verify that your card has the right capabilities. See…

http://www.nvidia.com/object/cuda_gpus.html

Is my card working?
If you downloaded the GPU Computing SDK code samples (see installation section above), then you can do a basic query of your card to make sure it is working and view its capabilities. From the Command Prompt window you can execute the following

```
C:\ProgramData\NVIDIA Corporation\CUDA
Samples\v5.5\bin\win64\Release\deviceQuery.exe Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce 8800 Ultra"
  CUDA Driver Version / Runtime Version          5.5 / 5.5
  CUDA Capability Major/Minor version number:    1.0
  Total amount of global memory:                 768 MBytes
(805306368 bytes)
  (16) Multiprocessors x (  8) CUDA Cores/MP:    128 CUDA Cores
  GPU Clock rate:                                1512 MHz (1.51
GHz)
  Memory Clock rate:                             1080 Mhz
  Memory Bus Width:                              384-bit
  Max Texture Dimension Size (x,y,z)             1D=(8192),
2D=(65536,32768), 3D=(2048,2048,2048)
  Max Layered Texture Size (dim) x layers        1D=(8192) x 512,
2D=(8192,8192) x 512
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       16384 bytes
  Total number of registers available per block: 8192
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  768
  Maximum number of threads per block:           512
  Maximum sizes of each dimension of a block:    512 x 512 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x
1
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             256 bytes
  Concurrent copy and kernel execution:          No with 0 copy
engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       No
```

```
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  CUDA Device Driver Mode (TCC or WDDM):         WDDM (Windows
Display Driver Model)
  Device supports Unified Addressing (UVA):      No
  Device PCI Bus ID / PCI location ID:           1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice()
with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 5.5,
CUDA Runtime Version = 5.5, NumDevs = 1, Device0
 = GeForce 8800 Ultra
```

Why doesn't GPU_RUN_TEST show a performance improvement?

If the **`gpu_run_test`** command indicates a negligible speedup (on the order of 1), and/or does not display any information about your graphics card, this may indicate that your IDL paths are not set correctly, or that gpulib is in emulation mode because there is no Cuda-enabled graphics card available. Check your paths (see "Step 4: Install GPULib"), restart your command prompt window, and retry.